

# A TRANSFORMATION APPROACH THAT MAKES SPAI, PSAI AND RSAI PROCEDURES EFFICIENT FOR LARGE DOUBLE IRREGULAR NONSYMMETRIC SPARSE LINEAR SYSTEMS\*

ZHONGXIAO JIA<sup>†</sup> AND WENJIE KANG<sup>‡</sup>

**Abstract.** It has been known that the sparse approximate inverse preconditioning procedures SPAI and PSAI(*tol*) are costly to construct preconditioners for a large sparse nonsymmetric linear system with the coefficient matrix having at least one relatively dense column. This is also true for SPAI and the recently proposed sparse approximate inverse preconditioning procedure RSAI(*tol*) procedure when the matrix has at least one relatively dense row. The matrix is called double irregular sparse if it has at least one relatively dense column and row, and it is double regular sparse if all the columns and rows of it are sparse. Double irregular sparse linear systems have a wide range of applications, and 24.4% of the nonsymmetric matrices in the Florida University collection are double irregular sparse. Making use of the Sherman-Morrison-Woodbury formula twice, we propose an approach that transforms a double irregular sparse problem into some double regular sparse ones with the same coefficient matrix, for which SPAI, PSAI(*tol*) and RSAI(*tol*) are efficient to construct preconditioners for the transformed double regular linear systems. We consider some theoretical and practical issues on such transformation approach, and develop a practical algorithm that first preconditions the transformed systems and then solves them by Krylov iterative solvers. Numerical experiments confirm the very sharp superiority of our transformation algorithm to SPAI, PSAI(*tol*) and RSAI(*tol*) applied to the double irregular sparse linear problem directly.

**Key words.** Sparse linear system, preconditioning, sparse approximate inverse, double irregular sparse, double regular sparse, Sherman-Morrison-Woodbury formula, F-norm minimization, Krylov solver

**AMS subject classifications.** 65F10

**1. Introduction.** In scientific and engineering computing, a core task is to solve the large sparse nonsymmetric linear system

$$(1.1) \quad Ax = b,$$

where  $A$  is an  $n \times n$  real nonsymmetric and nonsingular matrix, and  $b$  is a given  $n$ -dimensional vector. Krylov iterative solvers, such as the generalized minimal residual method (GMRES) and the biconjugate gradient stabilized method (BiCGStab) [28], have been commonly used for solving (1.1) in nowadays. However, the convergence of Krylov solvers are generally extremely slow when  $A$  is ill conditioned or has bad spectral property [28]. So it is critical to use preconditioning techniques to accelerate the convergence of Krylov solvers. Sparse approximate inverse (SAI) preconditioning procedures have been one class of the most important general-purpose preconditioning procedures [5, 14, 28]. Their goal is to construct a sparse approximate inverse  $M \approx A^{-1}$  directly.

There are two kinds of approaches to computing  $M$ . One of them gets a factorized  $M \approx M_1 M_2$ , and the preconditioned linear system is  $M_1 A M_2 y = M_1 b$ ,  $x = M_2 y$ . Typical algorithms of this kind are approximate inverse (AINV) type algorithms [6, 8] and the balanced incomplete factorization (BIF) algorithm [9, 10]. Stabilized and

---

\*Supported in part by National Science Foundation of China (No. 11371219).

<sup>†</sup>Department of Mathematical Sciences, Tsinghua University, Beijing, 100084, People's Republic of China, jiaxz@tsinghua.edu.cn.

<sup>‡</sup>Department of Mathematical Sciences, Tsinghua University, Beijing, 100084, People's Republic of China, kangwj11@mails.tsinghua.edu.cn.

block versions of the AINV factorized approximate inverse preconditioner are proposed in [7]. The other kind of approach is based on F-norm minimization, which is inherently parallelizable and computes a right preconditioner  $M$  by minimizing  $\|AM - I\|_F$  with certain sparsity constraints on  $M$ , where  $\|\cdot\|_F$  denotes the Frobenius norm of a matrix and  $I$  is the identity matrix of order  $n$ . The preconditioned linear system is  $AMy = b$ ,  $x = My$ . A key of this approach is the efficient determination of an effective sparsity pattern of  $A^{-1}$ . If the sparsity pattern of  $M$  is prescribed, the resulting procedure is called a static one. See, e.g., [3, 4, 11, 15, 18] for details. If the sparsity pattern of  $M$  is adaptively determined during the computational process, the procedure is called adaptive. The SPAI algorithm proposed by Grote and Huckle [17] has been a popular adaptive F-norm minimization based SAI preconditioning procedure. It has been generalized to block form, called BSPAI, in [2]. Jia and Zhu [26] have proposed an adaptive Power SAI (PSAI) preconditioning procedure and developed a practical  $\text{PSAI}(tol)$  algorithm that, during the loops, drops the nonzero entries in  $M$  whose sizes are below some tolerances  $tol$ . Jia and Zhang [24] have recently established a mathematical theory on dropping tolerances  $tol$  for all static F-norm minimization based SAI procedures and the adaptive  $\text{PSAI}(tol)$ . The numerical experiments in [25, 26] have demonstrated that  $\text{PSAI}(tol)$  is at least competitive with SPAI and can outperform SPAI substantially. Very recently, the authors of this paper have proposed a Residual based Sparse Approximate Inverse (RSAI) preconditioning procedure [23], which is different from the way used in SPAI and is based on only the *partial but dominant* rather than all information on the current residual. They have developed a practical  $\text{RSAI}(tol)$  algorithm with dropping strategies exploited.  $\text{RSAI}(tol)$  improves the computational efficiency of SPAI and meanwhile constructs effective preconditioners  $M$ . Kolotilina and Yeremin [27] have proposed a factorized sparse approximate inverse (FSAI) preconditioning procedure, which is a mixture of the above two kinds of SAI preconditioning procedures. FSAI has been generalized to block form, called BFSAI, in [21]. An adaptive algorithm which gets the sparsity pattern of the BFSAI preconditioner  $M$  can be found in [19, 20, 22]. For more on SAI preconditioning procedures, we refer the reader to [5, 14].

The term of 'irregular sparse matrix' was introduced in [25], where an irregular sparse  $A$  means that it has at least one relatively dense column, whose number of nonzero entries is substantially more than the average number of nonzero entries per column of  $A$ . There,  $A$  is numerically declared to be column irregular sparse if it has one column whose number of nonzero entries is at least  $10p$ , where  $p$  is the average number of nonzero entries per column of  $A$  [25]. Following this standard, it is reported in [25] that column irregular linear problems have a wide range of applications and 34% of the square matrices in the University of Florida sparse matrix collection [12] are column irregular sparse; see [25] for further information on where column irregular sparse matrices come from and how dense irregular columns are, etc. For a column irregular sparse  $A$ , Jia and Zhang [25] have shown that SPAI is costly and even impractical to find and add indices at each loop to update the sparsity pattern of the current  $M$ . Meanwhile, it has been shown [25, 26] that  $M$  by SPAI may be ineffective for preconditioning (1.1), while  $\text{PSAI}(tol)$ , though also costly, can get an effective  $M$  for  $A$  column irregular sparse.

In the same way, we declare  $A$  to be row irregular sparse if it has at least one relatively dense row. If  $A$  is both column and row irregular sparse, it is called double irregular sparse. Correspondingly, if all the columns and rows of  $A$  are sparse,  $A$  is called double regular sparse. Numerically, we define an irregular row using the

same standard as that of an irregular column. By this definition, a column irregular symmetric matrix is double irregular sparse. We have investigated all the real non-symmetric square matrices in the collection [12], which contains 775 matrices. We have found that 225 of them are column irregular sparse and 189 of these column irregular sparse matrices are double irregular sparse, that is, 24.4% of the nonsymmetric matrices in the collection are double irregular. This shows that double irregular sparse linear systems are quite common in practical applications.

As we have mentioned, Jia and Zhang [25] have shown why the SPAI and PSAI(*tol*) are costly when they are applied to column irregular sparse problems. Furthermore, their analysis has revealed that SPAI is also costly when applied to  $A^T$  for computing a left preconditioner for  $A$  column irregular sparse, that is, we compute  $M$  by minimizing  $\|A^T M^T - I\|_F$  with certain sparsity constraints on  $M$  and gets the left-preconditioned system  $MAx = Mb$ . As we will explain later, SPAI is costly for  $A$  row irregular sparse when computing a right preconditioner  $M$ . Numerical experiments have illustrated that RSAI(*tol*) improves the computational efficiency of SPAI substantially for column irregular sparse problems [23]. However, it is still costly for row irregular sparse problems, as we will see later. Row irregular sparse matrices may also impair the efficiency of PSAI(*tol*). Consequently, for double irregular sparse problems, SPAI, PSAI(*tol*) and RSAI(*tol*) are costly and even impractical to construct preconditioners  $M$ . How to efficiently solve double irregular sparse linear systems using Krylov solvers preconditioned by SPAI, PSAI(*tol*) and RSAI(*tol*) is, therefore, a very significant topic. This constitutes the work of our current paper.

In order to implement SPAI and PSAI(*tol*) efficiently and construct effective preconditioners for the column irregular sparse (1.1), making use of the Sherman-Morrison-Woodbury formula, Jia and Zhang [25] have proposed an approach that transforms a column irregular sparse problem into some column regular ones with the same coefficient matrix and multiple right-hand sides, so that SPAI and PSAI(*tol*) are efficient to implement for the regular sparse matrix, compared to them applied to the original column irregular sparse  $A$  directly. Then an approximate solution of the original system with a prescribed accuracy is recovered from those of regular ones with the accuracy determined by the prescribed one for the original system. The numerical experiments in [25] have indicated that such transformation approach speeds up the computational efficiency of SPAI and PSAI(*tol*) applied to the original problem very substantially and SPAI improves its preconditioning effectiveness essentially. However, this approach does not treat row irregular sparse linear systems, for which SPAI, PSAI(*tol*) and RSAI(*tol*) are still costly, as pointed out previously.

As is known [23, 25, 26], a common point of these three procedures is that they can construct preconditioners  $M$  efficiently for  $A$  double regular sparse, among of which PSAI(*tol*) is most effective and SPAI and RSAI(*tol*) are comparably effective for preconditioning double regular sparse linear systems. Based on this key observation, in the spirit of [25], we, for double irregular sparse problems, propose an approach that transforms the given double irregular sparse problem (1.1) into some double regular sparse ones with the same coefficient matrix. Then we apply SPAI, PSAI(*tol*) and RSAI(*tol*) to the double regular sparse systems. Finally, we recover the solution of (1.1) from those of the double regular sparse ones. Precisely, suppose that  $A$  has  $s_1$  and  $s_2$  relatively dense columns and rows, respectively. Then our approach splits  $A$  into the sum of a double regular sparse matrix  $\hat{A}$  and two matrices of low ranks  $s_1$  and  $s_2$ , respectively. We should remind that  $s_1$  and  $s_2$  must be very small relative to the order  $n$  of  $A$ , otherwise  $A$  is not a sparse matrix. By making use of

the Sherman-Morrison-Woodbury formula [16] twice, our approach transforms (1.1) into  $s_1 + s_2 + 1$  double regular linear systems with the same coefficient matrix  $\hat{A}$  and  $s_1 + s_2 + 1$  multiple right-hand sides. Finally, we can recover the solution of (1.1) from those of the  $s_1 + s_2 + 1$  linear systems. By considering several theoretical and practical issues, we develop a practical algorithm for (1.1) that is first preconditioned by SPAI, PSAI(*tol*) or RSAI(*tol*) and then solved by Krylov solvers. We show how to get an approximate solution of (1.1) with a prescribed accuracy  $\varepsilon$  from those of the  $s_1 + s_2 + 1$  double regular ones with the accuracy determined by the prescribed accuracy  $\varepsilon$  for (1.1). Numerical experiments will exhibit the very sharp superiority of our transformation algorithm to the one that first preconditions (1.1) by SPAI, PSAI(*tol*) or RSAI(*tol*) directly and then solves the preconditioned linear systems by Krylov solvers.

The paper is organized as follows. In Section 2, we briefly review SPAI, PSAI(*tol*) and RSAI(*tol*) procedures. In Section 3, we propose our transformation approach for solving the double irregular sparse problem (1.1). In Section 4, we consider some theoretical and practical issues to develop a practical transformation algorithm. In Section 5, we make numerical experiments on a number of real-world double irregular sparse problems, confirming the very sharp superiority of our transformation algorithm to the algorithm that preconditions (1.1) by SPAI, PSAI(*tol*) or RSAI(*tol*) directly and solves it by Krylov solvers. Finally, we conclude the paper in Section 6.

**2. The SPAI, PSAI(*tol*) and RSAI(*tol*) procedures.** For an F-norm minimization based SAI preconditioning procedure, we need to solve the constrained minimization problem

$$(2.1) \quad \min_{M \in \mathcal{M}} \|AM - I\|_F,$$

where  $\mathcal{M}$  is the set of matrices with a given sparsity pattern  $\mathcal{J}$ . Define  $\mathcal{M}_k$  as the set of  $n$ -dimensional vectors whose sparsity pattern is  $\mathcal{J}_k = \{i \mid (i, k) \in \mathcal{J}\}$ , and let  $M = (m_1, m_2, \dots, m_n)$ . Then (2.1) is divided into  $n$  independent constrained least squares (LS) problems

$$(2.2) \quad \min_{m_k \in \mathcal{M}_k} \|Am_k - e_k\|, \quad k = 1, 2, \dots, n,$$

where  $e_k$  is the  $k$ th column of  $I$ . Here and hereafter,  $\|\cdot\|$  denotes the 2-norm of a matrix or vector. For each  $k$ , denote by  $\mathcal{I}_k$  the set of indices of nonzero rows of  $A(\cdot, \mathcal{J}_k)$ , and define  $A_k = A(\mathcal{I}_k, \mathcal{J}_k)$ , the vectors  $\tilde{m}_k = m_k(\mathcal{J}_k)$  and  $\tilde{e}_k = e_k(\mathcal{I}_k)$ . Then (2.2) amounts to solving the smaller unconstrained LS problems

$$(2.3) \quad \min_{\tilde{m}_k} \|A_k \tilde{m}_k - \tilde{e}_k\|, \quad k = 1, 2, \dots, n,$$

which can be solved by QR decompositions in parallel.

If  $M$  is not yet good enough, that is, at least one residual norm (2.2) for some  $k$  does not drop below a prescribed tolerance  $\eta$ , an adaptive SAI preconditioning procedure, such as SPAI [17], PSAI(*tol*) [26] and RSAI(*tol*) [23], refines it by augmenting or adjusting the sparsity pattern  $\mathcal{J}_k$  dynamically and updating  $\tilde{m}_k$ . We highlight that the unique fundamental mathematical distinction of these adaptive F-norm minimization based SAI preconditioning procedures is the way that augments or adjusts the sparsity pattern of  $M$ . It has been shown in [25, 26] that PSAI(*tol*) captures the sparsity pattern of  $A^{-1}$  more effectively than SPAI. In [23], we have shown that

RSAI(*tol*) and SPAI compute comparably effective preconditioners but the former is more efficient than the latter. In what follows we briefly review SPAI, PSAI(*tol*) and RSAI(*tol*).

**2.1. The SPAI procedure.** Denote by  $\mathcal{J}_k^{(l)}$  the sparsity pattern of  $m_k$  after  $l$  loops starting with an initial pattern  $\mathcal{J}_k^{(0)}$ , and define  $\mathcal{I}_k^{(l)}$  to be the set of indices of all nonzero rows of  $A(\cdot, \mathcal{J}_k^{(l)})$ . Let  $A_k = A(\mathcal{I}_k^{(l)}, \mathcal{J}_k^{(l)})$ ,  $\tilde{e}_k = e_k(\mathcal{I}_k^{(l)})$ , and  $\tilde{m}_k = m_k(\mathcal{J}_k^{(l)})$  be the solution of (2.3). Denote the residual of (2.2) by

$$(2.4) \quad r_k = Am_k - e_k.$$

If  $\|r_k\| = \|A_k\tilde{m}_k - \tilde{e}_k\| > \eta$ , denote by  $\mathcal{L}_k$  the set of indices  $i$  for which  $r_k(i) \neq 0$  and  $\mathcal{N}_k$  the set of indices of nonzero columns of  $A(\mathcal{L}_k, \cdot)$ . Then

$$(2.5) \quad \hat{\mathcal{J}}_k = \mathcal{N}_k \setminus \mathcal{J}_k^{(l)}$$

forms the new candidates for augmenting  $\mathcal{J}_k^{(l)}$  in the next loop of SPAI, in which  $\mathcal{J}_k^{(l)}$  is updated as follows [17]: For each  $j \in \hat{\mathcal{J}}_k$ , consider the one-dimensional minimization problem

$$(2.6) \quad \min_{\mu} \|r_k + \mu Ae_j\|,$$

whose solution is

$$(2.7) \quad \mu_j = -\frac{r_k^T Ae_j}{\|Ae_j\|^2},$$

and the 2-norm  $\rho_j$  of the new residual  $r_k + \mu_j Ae_j$  satisfies

$$(2.8) \quad \rho_j^2 = \|r_k\|^2 - \frac{(r_k^T Ae_j)^2}{\|Ae_j\|^2}.$$

SPAI selects a few, say  $1 \sim 5$ , most profitable indices from  $\hat{\mathcal{J}}_k$  with the smallest  $\rho_j$  and adds them to  $\mathcal{J}_k^{(l)}$  to obtain  $\mathcal{J}_k^{(l+1)}$ . Define  $\hat{\mathcal{I}}_k$  to be the set of indices of new nonzero rows related to the most profitable indices added, and let  $\mathcal{I}_k^{(l+1)} = \mathcal{I}_k^{(l)} \cup \hat{\mathcal{I}}_k$ . Then we solve the new LS problem

$$(2.9) \quad \min \|A(\mathcal{I}_k^{(l+1)}, \mathcal{J}_k^{(l+1)})m_k(\mathcal{J}_k^{(l+1)}) - e_k(\mathcal{I}_k^{(l+1)})\|$$

whose solution can be updated from the previous  $\tilde{m}_k$  efficiently. Proceed in such a way until  $\|r_k\| \leq \eta$  or  $l$  reaches the prescribed maximum  $l_{\max}$ , where  $\eta$  is a mildly small tolerance, usually,  $0.1 \sim 0.4$ ; see [17, 25, 26]. Obviously, if the cardinality of  $\hat{\mathcal{J}}_k$  is very big, SPAI is costly, which corresponds to the case that the  $k$ th column of  $A$  is relatively dense [25, 26]. Moreover, it is easy to find that SPAI is costly when an index in  $\mathcal{L}_k$  corresponds to a relatively dense row of  $A$ , which results in a very big cardinality of  $\hat{\mathcal{J}}_k$ . Therefore, SPAI is costly and even impractical for  $A$  double irregular sparse.

**2.2. The PSAI(*tol*) procedure.** The basic PSAI (BPSAI) procedure proposed in [26] is motivated by the Cayley–Hamilton theorem:  $A^{-1}$  can be expressed as

$$(2.10) \quad A^{-1} = \sum_{i=0}^{n-1} c_i A^i$$

where  $A^0 = I$  and the  $c_i$  are certain constants for  $i = 1, 2, 3, \dots, n-1$ . Denote by  $\mathcal{P}(\cdot)$  the sparsity pattern of a matrix or vector, and define the matrix  $|A| = (|a_{ij}|)$ . We see from (2.10) that  $\mathcal{P}(A^{-1}) \subseteq \mathcal{P}((I + |A|)^{n-1})$ . So the pattern  $\mathcal{J}$  of a good sparse approximate inverse  $M$  can be taken as a subset of  $\mathcal{P}((I + |A|)^{l_{\max}})$  for a given small  $l_{\max}$  in BPSAI. Thus, the sparsity pattern  $\mathcal{J}_k$  of the  $k$ th column of  $M$  is a subset of  $\bigcup_{i=0}^{l_{\max}} \mathcal{P}(|A|^i e_k)$  since  $\mathcal{P}((I + |A|)^{l_{\max}}) \subseteq \bigcup_{i=0}^{l_{\max}} \mathcal{P}(|A|^i)$ .

For  $k = 1, 2, \dots, n$ , BPSAI updates the sparsity pattern  $\mathcal{J}_k$  adaptively in the following way: For  $l = 0, 1, \dots, l_{\max}$ , denote by  $\mathcal{J}_k^{(l)}$  the sparsity pattern of  $m_k$  at loop  $l$  and by  $\mathcal{I}_k^{(l)}$  the set of nonzero row indices of  $A(\cdot, \mathcal{J}_k^{(l)})$ . Define  $a_k^{(l+1)} = Aa_k^{(l)}$  with  $a_k^{(0)} = e_k$ . Then the sparsity pattern  $\mathcal{J}_k^{(l+1)} = \mathcal{J}_k^{(l)} \cup \mathcal{P}(a_k^{(l+1)})$ . Denote by  $\hat{\mathcal{I}}_k$  the set of indices of new nonzero rows  $A(\cdot, \mathcal{J}_k^{(l+1)})$ , and let  $\mathcal{I}_k^{(l+1)} = \mathcal{I}_k^{(l)} \cup \hat{\mathcal{I}}_k$ . Then the new LS problem (2.3) can be solved by updating  $\tilde{m}_k$  instead of resolving it. Proceed in such a way until  $\|r_k\| \leq \eta$  or  $l > l_{\max}$ .

In order to develop a practical algorithm, we must control the sparsity of  $M$ . Jia and Zhu [26] have proposed the PSAT(tol) algorithm, which, during the loops, drops those nonzero entries whose magnitudes are below a prescribed threshold  $tol$  and retains only those large ones. It has turned out that  $tol$  has strong effects on the preconditioning quality of  $M$ . Jia and Zhang [24] have established a mathematical theory on robust dropping tolerances for PSAT(tol) and all the static F-norm minimization based SAI preconditioning procedures, by which they have devised an adaptive and robust dropping criterion: at loop  $l \leq l_{\max}$ , a nonzero entry  $m_{jk}$  is dropped if

$$(2.11) \quad |m_{jk}| \leq \frac{\eta}{nnz(m_k)\|A\|_1}, j = 1, 2, \dots, n,$$

where  $nnz(m_k)$  is the number of nonzero entries in  $m_k$  and  $\|\cdot\|_1$  is the 1-norm of a matrix. From the theory in [24], PSAT(tol) with the above dropping criterion will get the preconditioner  $M$  that is as sparse as possible and meanwhile has comparable preconditioning quality to the one generated by BPSAI without dropping any entries. However, PSAT(tol) involves large sized LS problems for column irregular sparse matrices, though  $M$  obtained by it is an effective preconditioner [25]. As a whole, if  $A$  is double irregular sparse, PSAT(tol) is costly and may be impractical.

**2.3. The RSAI(tol) procedure.** We first overview the basic RSAI (BRSAT) procedure [23]. Suppose that  $\mathcal{J}_k^{(l)}$  is the sparsity pattern of  $m_k$  generated by BRSAT after  $l$  loops starting with the initial sparsity pattern  $\mathcal{J}_k^{(0)}$ . If  $m_k$  does not yet satisfy the prescribed accuracy  $\eta$ , BRSAT continues improving  $m_k$  by augmenting its sparsity pattern as follows.

Denote by  $\mathcal{L}_k$  the set of indices  $i$  for which  $r_k(i) \neq 0$ . BRSAT takes precedence to reduce those largest entries in  $r_k$  since they contribute most to the size of  $\|r_k\|$ . We call the indices corresponding to the largest entries of  $r_k$  the dominant indices. Denote by  $\hat{\mathcal{R}}_k^{(l)}$  the set of the dominant indices  $i$  with several largest  $|r_k(i)|$ . Define  $\hat{\mathcal{J}}_k$  to be the set of all new column indices of  $A$  that correspond to  $\hat{\mathcal{R}}_k^{(l)}$  but do not appear in  $\mathcal{J}_k^{(l)}$ . Then we set

$$\mathcal{J}_k^{(l+1)} = \mathcal{J}_k^{(l)} \cup \hat{\mathcal{J}}_k.$$

When choosing  $\hat{\mathcal{R}}_k^{(l)}$  from  $\mathcal{L}_k$  in the above way, it may happen that  $\hat{\mathcal{R}}_k^{(l)} = \hat{\mathcal{R}}_k^{(l-1)}$ . If



so, set

$$\mathcal{R}_k^{(l)} = \bigcup_{i=0}^{l-1} \hat{\mathcal{R}}_k^{(i)},$$

and choose  $\hat{\mathcal{R}}_k^{(l)}$  from the set whose elements are in  $\mathcal{L}_k$  but not in  $\mathcal{R}_k^{(l)}$ . In this way,  $\hat{\mathcal{R}}_k^{(l)}$  is always non-empty unless  $m_k$  is exactly the  $k$ th column of  $A^{-1}$  [23]. Denote  $\hat{\mathcal{I}}_k$  by the set of indices of new nonzero rows corresponding to the added column indices  $\hat{\mathcal{J}}_k$ . Then we update  $\mathcal{I}_k^{(l+1)} = \mathcal{I}_k^{(l)} \cup \hat{\mathcal{I}}_k$  and solve the new LS problem (2.9), generating a better approximation  $m_k$  to the  $k$ th column of  $A^{-1}$ . Repeat this process for  $k = 1, 2, \dots, n$  until  $\|r_k\| \leq \eta$  or  $l$  exceeds  $l_{\max}$ .

Similar to the PSAI(*tol*) algorithm, in order to control the sparsity of  $M$ , a practical RSAI(*tol*) algorithm has been developed in [23] that introduces the dropping criterion (2.11) into BRSAI. It has been shown in [23] why RSAI(*tol*) is as equally effective as SPAI, but it is more efficient than SPAI for  $A$  column irregular sparse because we use only a few other than all indices of those nonzero  $r_k(i)$  and do not compute those numbers  $\rho_j$  in SPAI, avoiding sorting them and picking up the most profitable indices. However, RSAI(*tol*) may be costly for  $A$  is row irregular sparse. The reason is as follows: Suppose that the  $k$ th row of  $A$  is relatively dense. Then once  $\{k\} \subseteq \hat{\mathcal{R}}_k^{(l)}$ , the resulting  $m_k$  is also relatively dense, leading to a relatively large sized (2.3). Therefore, if  $A$  is double irregular sparse, RSAI(*tol*) is still costly and may be impractical.

**3. Transformation of double irregular sparse linear systems into regular ones.** The previous discussion has indicated that SPAI, PSAI(*tol*) and RSAI(*tol*) is costly and even impractical when  $A$  is double irregular sparse. In order to improve their efficiency, motivated by the approach in [25], we attempt to transform a double irregular sparse (1.1) into some double regular sparse ones, for which SPAI, PSAI(*tol*) and RSAI(*tol*) are efficient for constructing possibly effective  $M$ . Such approach is based on the following well-known Sherman-Morrison-Woodbury formula [16, p. 50]).

LEMMA 3.1. *Let  $U, V \in R^{n \times s}$  with  $s \leq n$ . If  $A$  is nonsingular, then  $A + UV^T$  is nonsingular if and only if  $I + V^T A^{-1} U$  is nonsingular. Furthermore,*

$$(3.1) \quad (A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1}.$$

In practical applications, one is typically interested in the formula for  $s \ll n$ , which reduces to the Sherman-Morrison formula when  $s = 1$ .

In what follows, for a double irregular sparse  $A$  we assume that the  $j_1, j_2, \dots, j_{s_1}$ th columns and  $i_1, i_2, \dots, i_{s_2}$ th rows of  $A$  are relatively dense, respectively. Let  $A_{dc} = (a_{j_1}, a_{j_2}, \dots, a_{j_{s_1}})$ , where  $a_{j_k}$  is the  $j_k$ th column of  $A$ , and  $\tilde{A}_{dc} = (\tilde{a}_{j_1}, \tilde{a}_{j_2}, \dots, \tilde{a}_{j_{s_1}})$  the sparsification of  $A_{dc}$ , each column of which is sparse and retains only a few nonzero entries of  $a_{j_k}$ . Let  $U_1 = A_{dc} - \tilde{A}_{dc} = (u_1, u_2, \dots, u_{s_1})$ . It is obvious that the nonzero entries of  $U_1$  are just those dropped ones of  $A_{dc}$ . Define  $\tilde{A}$  to be the matrix that replaces the dense columns  $a_{j_k}$  of  $A$  by the sparse vectors  $\tilde{a}_{j_k}$ ,  $k = 1, 2, \dots, s_1$ . Then  $\tilde{A}$  is column regular sparse and is related to  $A$  by

$$(3.2) \quad A = \tilde{A} + U_1 V_1^T,$$

where  $V_1 = (e_{j_1}, e_{j_2}, \dots, e_{j_{s_1}})$  with  $e_{j_k}$  the  $j_k$ th column of  $I$ . Assume that  $\tilde{A}$  and  $I + V_1^T \tilde{A}^{-1} U_1$  are nonsingular. By (3.1), we have

$$(3.3) \quad A^{-1} = \tilde{A}^{-1} - \tilde{A}^{-1} U_1 (I + V_1^T \tilde{A}^{-1} U_1)^{-1} V_1^T \tilde{A}^{-1}.$$

Clearly, the  $i_1, i_2, \dots, i_{s_2}$ th rows of  $\tilde{A}$  are still dense. Next, we further transform  $\tilde{A}$  into double regular sparse. Let  $\tilde{A}_{dr} = (\tilde{a}_{i_1}, \tilde{a}_{i_2}, \dots, \tilde{a}_{i_{s_2}})^T$ , where  $\tilde{a}_{i_k}^T$  is the  $i_k$ th dense row of  $\tilde{A}$ ,  $k = 1, 2, \dots, s_2$ . Define  $\hat{A}_{dr} = (\hat{a}_{i_1}, \hat{a}_{i_2}, \dots, \hat{a}_{i_{s_2}})^T$  to be the sparsification of  $\tilde{A}_{dr}$ , where each row  $\hat{a}_{i_k}^T$  is sparse and retains only a few nonzero entries of  $\tilde{a}_{i_k}^T$ . Let  $V_2 = \tilde{A}_{dr}^T - \hat{A}_{dr}^T = (v_1, v_2, \dots, v_{s_2})$ . Then the nonzero entries of  $V_2$  are just those dropped ones of  $\tilde{A}_{dr}$ . Let  $\hat{A}$  be the matrix that replaces the dense rows  $\tilde{a}_{i_k}^T$  of  $\tilde{A}$  by the sparse vectors  $\hat{a}_{i_k}^T$ ,  $k = 1, 2, \dots, s_2$ . Then  $\hat{A}$  is double regular sparse and is related to  $\tilde{A}$  by

$$(3.4) \quad \tilde{A} = \hat{A} + U_2 V_2^T,$$

where  $U_2 = (e_{i_1}, e_{i_2}, \dots, e_{i_{s_2}})$  with  $e_{i_k}$  the  $i_k$ th column of  $I$ . Assume that  $\hat{A}$  and  $I + V_2^T \hat{A}^{-1} U_2$  are nonsingular. By (3.1), we have

$$(3.5) \quad \tilde{A}^{-1} = \hat{A}^{-1} - \hat{A}^{-1} U_2 (I + V_2^T \hat{A}^{-1} U_2)^{-1} V_2^T \hat{A}^{-1}.$$

From (3.3), the solution of (1.1) is

$$(3.6) \quad x = A^{-1} b = \tilde{A}^{-1} b - \tilde{A}^{-1} U_1 (I + V_1^T \tilde{A}^{-1} U_1)^{-1} V_1^T \tilde{A}^{-1} b.$$

Define  $y = \tilde{A}^{-1} b$  and  $W = \tilde{A}^{-1} U_1$ . Substituting (3.5) into (3.6), we obtain

$$(3.7) \quad y = \tilde{A}^{-1} b = \hat{A}^{-1} b - \hat{A}^{-1} U_2 (I + V_2^T \hat{A}^{-1} U_2)^{-1} V_2^T \hat{A}^{-1} b$$

and

$$(3.8) \quad W = \tilde{A}^{-1} U_1 = \hat{A}^{-1} U_1 - \hat{A}^{-1} U_2 (I + V_2^T \hat{A}^{-1} U_2)^{-1} V_2^T \hat{A}^{-1} U_1.$$

Keeping in mind the definitions of  $U_1$  and  $U_2$ , we see from the above relations that the computation of  $y$  and  $W$  involves the solutions of the double regular sparse linear system

$$(3.9) \quad \hat{A} z = b,$$

the  $s_1$  double regular sparse linear systems

$$(3.10) \quad \hat{A} p = u_k, k = 1, 2, \dots, s_1$$

and the other  $s_2$  double regular sparse linear systems

$$(3.11) \quad \hat{A} q = e_{i_k}, k = 1, 2, \dots, s_2.$$

With  $y$  and  $W$  at hand, we then get the solution of  $Ax = b$  from (3.6).

The above derivation shows that  $\hat{A} = A - U_1 V_1^T - U_2 V_2^T$ , indicating that  $\hat{A}$  is the sum of  $A$  plus rank  $s_1$  and  $s_2$  modifications. We can now summarize the above transformation approach to solving  $Ax = b$  as Algorithm 1. In our context, its implementation consists of four steps: Firstly, we construct the double regular sparse



---

**Algorithm 1** Solving the double irregular sparse linear system (1.1)

---

- 1: Find  $s_1$  and  $A_{dc}$ , and sparsify  $A_{dc}$  to obtain  $\tilde{A}_{dc}$ . Let  $U_1 = A_{dc} - \tilde{A}_{dc} = (u_1, u_2, \dots, u_{s_1})$  and  $V_1 = (e_{j_1}, e_{j_2}, \dots, e_{j_{s_1}})$ , and define  $\tilde{A} = A - U_1 V_1^T$ .
- 2: Find  $s_2$  and  $\tilde{A}_{dr}$ , and sparsify  $\tilde{A}_{dr}$  to obtain  $\hat{A}_{dr}$ . Let  $V_2 = \tilde{A}_{dr}^T - \hat{A}_{dr}^T = (v_1, v_2, \dots, v_{s_2})$  and  $U_2 = (e_{i_1}, e_{i_2}, \dots, e_{i_{s_2}})$ , and define  $\hat{A} = \tilde{A} - U_2 V_2^T$ .
- 3: Solve the  $s_1 + s_2 + 1$  linear systems (3.9), (3.10) and (3.11) for  $z, p_1, p_2, \dots, p_{s_1}$  and  $q_1, q_2, \dots, q_{s_2}$ , respectively. Let  $P = (p_1, p_2, \dots, p_{s_1})$  and  $Q = (q_1, q_2, \dots, q_{s_2})$ . Compute  $y$  by

$$(3.12) \quad y = \tilde{A}^{-1}b = z - Q(I + V_2^T Q)^{-1}(V_2^T z)$$

and  $W$  by

$$(3.13) \quad W = P - Q(I + V_2^T Q)^{-1}(V_2^T P).$$

- 4: Compute the solution  $x$  of  $Ax = b$  by

$$(3.14) \quad x = A^{-1}b = y - W(I + V_1^T W)^{-1}(V_1^T y).$$


---

$\tilde{A}$ ; secondly, we compute sparse approximate inverses  $M$  of  $\hat{A}$  by SPAI, PSAI(*tol*) or RSAI(*tol*) and use them as preconditioners for the  $s_1 + s_2 + 1$  double regular sparse linear systems; thirdly, we solve the preconditioned systems iteratively by Krylov solvers; finally, we recover the solution of (1.1) from the  $s_1 + s_2 + 1$  solutions of the double regular sparse linear systems. The three procedures SPAI, PSAI(*tol*) and RSAI(*tol*) are now expected to be much more efficient than them applied to  $A$  directly. On the other hand, as argued in [25, 26], as preconditioners, the  $M$  constructed by the three procedures applied to  $\hat{A}$  are at least as effective as the corresponding ones applied to  $A$ . In addition, it is known [1, 5] that the setup time of  $M$  generally overwhelms that of Krylov iterations of the preconditioned system even in a parallel computing environment, provided that  $M$  is effective for preconditioning. In view of these, it is expected that the overall efficiency of Algorithm 1 is improved greatly, compared to the standard algorithm that first preconditions (1.1) by one of SPAI, PSAI(*tol*) or RSAI(*tol*) directly and then solves it by Krylov solvers.

**4. Theoretical analysis and practical considerations.** In this section, we will give some theoretical analysis and practical considerations on the proposed transformation approach, by which we are based on Algorithm 1 to develop a practical and reliable iterative solver for (1.1). To this end, we need to consider several issues.

First of all, we adapt the definition of column irregular sparse in [25] to double irregular sparse. We declare a column or row is dense if the number of nonzero entries in it exceeds  $10p$ , where  $p = \lfloor \frac{nnz(A)}{n} \rfloor$  is the average number of nonzero entries per column or row of  $A$ . The second issue is which nonzero entries in  $A_{dc}$  and  $\tilde{A}_{dr}$  should be dropped to generate  $\tilde{A}$  and  $\hat{A}$ , respectively. We adopt the same strategy as that in [25] and retain the diagonal and the  $p - 1$  nonzero entries nearest to the diagonal in each column of  $A_{dc}$ , so do  $p$  nonzero entries in each row of  $\tilde{A}_{dr}$ , supposing that all the diagonals of  $A$  are nonzero.

As we have seen, when transforming the double irregular sparse (1.1) into the

double regular sparse ones, we always assume that the column regular sparse  $\tilde{A}$  and the double regular sparse  $\hat{A}$  are nonsingular. Our third issue is: for which classes of matrices  $A$ , the nonsingularity of  $\hat{A}$  is ensured. Jia and Zhang [25] have investigated this problem in some detail for the column regular sparse  $\tilde{A}$  and established a number of results, which are directly adapted to our current double regular sparse  $\hat{A}$ , as Theorem 4.1 states.

**THEOREM 4.1.**  *$\hat{A}$  obtained by the above transformation approach is nonsingular for the following classes of matrices:*

- (i)  *$A$  is strictly row or column diagonally dominant.*
- (ii)  *$A$  is irreducibly row or column diagonally dominant.*
- (iii)  *$A$  is an  $M$ -matrix.*
- (iv)  *$A$  is an  $H$ -matrix.*

The above four classes of matrices have wide applications. Similar to [25], we can extend the first two classes of matrices in Theorem 4.1 to more general forms.

**COROLLARY 4.2.**  *$\hat{A}$  obtained by the above transformation approach is nonsingular for the following classes of matrices:*

- (i)  *$AD$  or  $DA$  is strictly row or column diagonally dominant where  $D$  is an arbitrary nonsingular diagonal matrix.*
- (ii)  *$PAQ$  is strictly row or column diagonally dominant where  $P$  and  $Q$  are permutation matrices.*
- (iii)  *$(PAQ)D$  or  $D(PAQ)$  is strictly row or column diagonally dominant where  $D$  is an arbitrary nonsingular diagonal matrix and  $P, Q$  are permutation matrices.*
- (iv)  *$A$  is an irreducible analogue of the matrices in (i)–(iii).*

There should exist more classes of matrices for which the resulting  $\hat{A}$  are nonsingular. We do not pursue this topic further. Strikingly, we will numerically find that for a general nonsingular  $A$  that does not belong to the aforementioned classes of matrices, the resulting  $\hat{A}$  is indeed nonsingular. This fact has been extensively verified for column irregular sparse matrices and column regular sparse  $\tilde{A}$  [25].

The fourth issue is the conditioning of  $\hat{A}$ . This is a hard problem. Theoretically, for a general nonsingular  $A$ , the conditioning of  $\hat{A}$  may become better or worse. When  $A$  is strictly row or column diagonally dominant or  $M$ -matrix, Jia and Zhang [25] have given mathematical justifications on why the resulting column regular sparse  $\tilde{A}$  is generally better conditioned than  $A$ . The same arguments works for these matrices in our current context, and  $\hat{A}$  can be shown to be generally better conditioned than  $A$ . Remarkably, later numerical experiments will indicate that the resulting  $\hat{A}$  is always and often considerably better conditioned than  $A$  for a general double irregular sparse  $A$  that does not fall into the matrices in Theorem 4.1 and Corollary 4.2.

The above considerations on the third and fourth issues indicates that our transformation approach is of generality in practical applications.

The final issue is the selection of stopping criteria for Krylov iterative solvers applied to the  $s_1 + s_2 + 1$  double regular sparse linear systems, given the prescribed accuracy  $\varepsilon$  for (1.1). This selection is crucial for reliably recovering an approximate solution of (1.1) with the prescribed accuracy  $\varepsilon$ . In order to solve this problem, we present the following theorem, based on which we can design reliable stopping criteria for the double regular linear systems.

**THEOREM 4.3.** *For  $U_1, U_2, V_1$  and  $V_2$  defined in Algorithm 1, let  $\tilde{z}, \tilde{p}_j, j = 1, 2, \dots, s_1$  and  $\tilde{q}_j, j = 1, 2, \dots, s_2$ , be the approximate solutions of (3.9), (3.10) and (3.11), respectively. Define  $\tilde{P} = (\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_{s_1})$ ,  $\tilde{Q} = (\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_{s_2})$  and the residuals  $r_{\tilde{z}} = b - \hat{A}\tilde{z}$ ,  $r_{\tilde{p}_j} = u_j - \hat{A}\tilde{p}_j$ ,  $r_{\tilde{q}_j} = e_{i_j} - \hat{A}\tilde{q}_j$ . Assume that  $I + V_2^T \tilde{Q}$  is*

nonsingular, and define  $c_1 = \|(I + V_2^T \check{Q})^{-1} V_2^T \check{z}\|$  and  $c_2 = \|(I + V_2^T \check{Q})^{-1} V_2^T \check{P}\|$ . Take

$$(4.1) \quad \check{y} = \check{z} - \check{Q}(I + V_2^T \check{Q})^{-1} V_2^T \check{z}$$

and

$$(4.2) \quad \check{W} = \check{P} - \check{Q}(I + V_2^T \check{Q})^{-1} V_2^T \check{P}$$

to be the approximations of  $y$  and  $W$  defined by (3.7) and (3.8), respectively, assume that  $I + V_1^T \check{W}$  is nonsingular, and define  $c_0 = \|(I + V_1^T \check{W})^{-1} V_1^T \check{y}\|$  and

$$(4.3) \quad \check{x} = \check{y} - \check{W}(I + V_1^T \check{W})^{-1} V_1^T \check{y}$$

to be an approximate solution of  $Ax = b$ . Then if

$$(4.4) \quad \frac{\|r_{\check{z}}\|}{\|b\|} \leq \frac{\varepsilon}{4},$$

$$(4.5) \quad \frac{\|r_{\check{p}_j}\|}{\|u_j\|} \leq \frac{\|b\|}{4\sqrt{s_1}c_0\|u_j\|}\varepsilon, j = 1, 2, \dots, s_1$$

and

$$(4.6) \quad \frac{\|r_{\check{q}_j}\|}{\|e_{ij}\|} \leq \frac{\|b\|}{2\sqrt{s_2}(c_0c_2 + c_1)}\varepsilon, j = 1, 2, \dots, s_2,$$

we have

$$(4.7) \quad \frac{\|r\|}{\|b\|} = \frac{\|b - A\check{x}\|}{\|b\|} \leq \varepsilon.$$

*Proof.* Define  $R_{\check{W}} = U_1 - \tilde{A}\check{W}$  and  $r_{\check{y}} = b - \tilde{A}\check{y}$ . From (3.2), we get

$$\begin{aligned} r &= b - A\check{x} = b - (\tilde{A} + U_1 V_1^T)(\check{y} - \check{W}(I + V_1^T \check{W})^{-1} V_1^T \check{y}) \\ &= b - \tilde{A}\check{y} - U_1 V_1^T \check{y} + \tilde{A}\check{W}(I + V_1^T \check{W})^{-1} V_1^T \check{y} + U_1 V_1^T \check{W}(I + V_1^T \check{W})^{-1} V_1^T \check{y} \\ &= r_{\check{y}} - U_1[(I + V_1^T \check{W}) - V_1^T \check{W}](I + V_1^T \check{W})^{-1} V_1^T \check{y} + \tilde{A}\check{W}(I + V_1^T \check{W})^{-1} V_1^T \check{y} \\ &= r_{\check{y}} - U_1(I + V_1^T \check{W})^{-1} V_1^T \check{y} + \tilde{A}\check{W}(I + V_1^T \check{W})^{-1} V_1^T \check{y} \\ (4.8) \quad &= r_{\check{y}} - R_{\check{W}}(I + V_1^T \check{W})^{-1} V_1^T \check{y}. \end{aligned}$$

Define  $R_{\check{P}} = U_1 - \hat{A}\check{P}$  and  $R_{\check{Q}} = U_2 - \hat{A}\check{Q}$ . From (4.1) and (4.2) we obtain

$$(4.9) \quad r_{\check{y}} = r_{\check{z}} - R_{\check{Q}}(I + V_2^T \check{Q})^{-1} V_2^T \check{z}$$

and

$$(4.10) \quad R_{\check{W}} = R_{\check{P}} - R_{\check{Q}}(I + V_2^T \check{Q})^{-1} V_2^T \check{P},$$

respectively. From (4.8), (4.9) and (4.10), by the definitions of  $c_0$ ,  $c_1$  and  $c_2$  we have

$$\begin{aligned} (4.11) \quad \|r\| &\leq \|r_{\check{y}}\| + c_0\|R_{\check{W}}\| \leq \|r_{\check{z}}\| + c_1\|R_{\check{Q}}\| + c_0(\|R_{\check{P}}\| + c_2\|R_{\check{Q}}\|) \\ &\leq \|r_{\check{z}}\| + c_0\|R_{\check{P}}\|_F + (c_0c_2 + c_1)\|R_{\check{Q}}\|_F. \end{aligned}$$

Noting that  $R_{\tilde{P}}(:, j) = r_{\tilde{p}_j}$  and  $R_{\tilde{Q}}(:, j) = r_{\tilde{q}_j}$ , we get  $\|R_{\tilde{P}}\|_F = \sqrt{\sum_{j=1}^{s_1} \|r_{\tilde{p}_j}\|^2}$  and  $\|R_{\tilde{Q}}\|_F = \sqrt{\sum_{j=1}^{s_2} \|r_{\tilde{q}_j}\|^2}$ . If (4.4) is satisfied and

$$\frac{\|r_{\tilde{p}_j}\|}{\|b\|} \leq \frac{\varepsilon}{4\sqrt{s_1}c_0}, j = 1, 2, \dots, s_1$$

and

$$\frac{\|r_{\tilde{q}_j}\|}{\|b\|} \leq \frac{\varepsilon}{2\sqrt{s_2}(c_0c_2 + c_1)}, j = 1, 2, \dots, s_2,$$

which are just (4.5) and (4.6), respectively, it follows from (4.11) that (4.7) holds.  $\square$

While Theorem 4.3 gives the stopping criteria for the double regular sparse problems, they are not directly applicable for the given prescribed accuracy  $\varepsilon$  of (1.1). The reason is that we cannot compute  $c_0$ ,  $c_1$  and  $c_2$  until iterations for (3.9), (3.10) and (3.11) terminate, but the stopping criteria (4.5) and (4.6) for (3.10) and (3.11) depend on  $c_0$ ,  $c_1$  and  $c_2$ . As a result, the computation of  $c_0$ ,  $c_1$  and  $c_2$  and the termination of iterative solvers interact, so that we cannot get  $c_0$ ,  $c_1$  and  $c_2$  in advance. Fortunately, it appears that their accurate estimates are not necessary and quite rough replacements are enough. It is seen that  $c_0$  is moderate if  $I + V_1^T \tilde{W}$  is not ill conditioned. Suppose that  $I + V_1^T \tilde{W}$  and  $I + V_2^T \tilde{Q}$  are not very ill conditioned. We observe that  $c_1$  and  $c_2$  rely on the norm of  $V_2$  and  $c_0$  depends on that of  $V_1$ . In implementations, we simply replace  $c_0$  by 1 and  $c_1$ ,  $c_2$  by  $\max_{1 \leq i \leq s_2} \|V_2(:, i)\|$ , respectively. In numerical experiments, we will find that such choices of  $c_0$ ,  $c_1$  and  $c_2$  work reliably and makes (4.7) hold for all the test problems.

Having done the above, we have finally developed Algorithm 1 into a truly working algorithm, called Algorithm 2 hereafter.

**5. Numerical experiments.** In this section, we test Algorithm 2 on a number of real-world nonsymmetric problems that are double irregular sparse, where we first construct sparse approximate inverses  $M$  of double regular sparse matrices  $\hat{A}$  by SPAI, PSAI(*tol*) or RSAI(*tol*) and then solve the resulting  $s_1 + s_2 + 1$  preconditioned double regular sparse linear systems by the Krylov solver BiCGStab, whose code is from Matlab 7.8.0. We will compare Algorithm 2 with the algorithm that preconditions (1.1) by SPAI, PSAI(*tol*) or RSAI(*tol*) directly and solves the preconditioned systems by BiCGStab. Depending on which of SPAI, PSAI(*tol*) and RSAI(*tol*) is used, Algorithm 2 gives rise to three algorithms, named new-SPAI, new-PSAI(*tol*) and new-RSAI(*tol*), abbreviated as N-SPAI, N-PSAI(*tol*) and N-RSAI(*tol*), respectively. Similarly, denote the algorithms that directly precondition (1.1) with SPAI, PSAI(*tol*) and RSAI(*tol*) by standard-SPAI, standard-PSAI(*tol*), and standard-RSAI(*tol*), denoted by S-SPAI, S-PSAI(*tol*), and S-RSAI(*tol*) for short, respectively. We shall demonstrate that N-SPAI, N-PSAI(*tol*) and N-RSAI(*tol*) outperform S-SPAI, S-PSAI(*tol*) and S-RSAI(*tol*) very greatly, respectively.

We perform numerical experiments on an Intel Core 2 Quad CPU E8400@3.00GHz with 2GB main memory using Matlab 7.8.0 with the machine precision  $\epsilon_{\text{mach}} = 2.22 \times 10^{-16}$  under the Linux operating system. PSAI(*tol*) and RSAI(*tol*) are experimental Matlab codes for the sequential environment. We use the SPAI 3.2 package for the SPAI algorithm, which is written in C/MPI [1] and is an optimized code in some sense. We take the initial sparsity pattern as that of  $I$  for SPAI and RSAI(*tol*). We apply row Dulmage-Mendelsohn permutations to the matrices having zero diagonals so as to make their diagonals nonzero [13]. The related Matlab commands

TABLE 5.1  
The description of test matrices

matrices	$n$	$nnz$	Description
rajat04	1041	8725	circuit simulation problem
rajat12	1879	12818	circuit simulation problem
rajat13	7598	48762	circuit simulation problem
memplus	17758	99147	computer component design memory circuit
ASIC_100k	99340	940621	Sandia, Xyce circuit simulation matrix
dc1	116835	766396	circuit simulation problem
dc2	116835	766396	circuit simulation problem
dc3	116835	766396	circuit simulation problem
trans4	116835	749800	circuit simulation problem
trans5	116835	749800	circuit simulation problem

are  $j = \text{demperm}(A(j, :))$ . We applied `demperm` to `rajat04`, `rajat12`, `rajat13` and `ASIC_100k`. We take  $c_0$ ,  $c_1$  and  $c_2$  to be the values defined in the end of Section 4. We stop BiCGStab when (4.4), (4.5) and (4.6) are satisfied with  $\varepsilon = 10^{-8}$  or 1000 iterations are used. The initial guess of the solution to each problem is always  $x_0 = 0$  and the right-hand side  $b$  is formed by choosing the solution  $x = (1, 1, \dots, 1)^T$ . We have found that BiCGStab without preconditioning did not converge for all the test problems within 1000 iterations except `rajat04`, `rajat12` and `rajat13`. With  $\tilde{x}$  defined by (4.3), we compute the actual relative residual norm

$$(5.1) \quad r_{actual} = \frac{\|b - A\tilde{x}\|}{\|b\|}$$

and compare it with the required accuracy  $\varepsilon = 10^{-8}$ .

In Table 5.2, we give some detailed information on  $A$  and  $\hat{A}$  constructed by Algorithm 2. From Table 5.2, we observe that all the test matrices have some almost fully dense columns and rows except `rajat04`, `rajat12`, `rajat13` and `memplus`. The table also indicates that  $s_1$  and  $s_2$  are very small relative to  $n$ . Sparsifying those dense columns and rows, we find that the number of nonzero entries in  $\hat{A}$  are considerably smaller than those in  $A$ . We remark that all the test matrices do not belong to the classes of matrices in Theorem 4.1. We observe that all the  $\hat{A}$  are better and can be considerably better conditioned than the corresponding  $A$ . This demonstrates that our transformation of  $A$  into  $\hat{A}$  is of practical generality that ensures not only the non-singularity of  $\hat{A}$  but also improves the conditioning of linear systems.

Now we look into the effective approximate sparsity pattern of  $A^{-1}$  and that of  $\hat{A}^{-1}$  obtained by our transformation approach. We aim to show that effective sparse approximate inverses of double irregular and regular sparse matrices are structure preserving, though theoretically good approximate inverses of a double regular sparse matrix may be irregular sparse [25]. We take `rajat04` as an example. Performing a row Dulmage-Mendelsohn permutation on it, we depict the sparsity patterns of  $A$  and  $\hat{A}$ . We then use the Matlab function `inv` to compute  $A^{-1}$  and  $\hat{A}^{-1}$  accurately. Then we drop their nonzero entries whose magnitudes fall below  $10^{-3}$ , and get good sparse approximate inverses of  $A$  and  $\hat{A}$ . Figure 5.1 depicts the patterns of  $A$ ,  $\hat{A}$ , and the sparsified  $A^{-1}$  and  $\hat{A}^{-1}$ . As we see clearly, the good approximate inverses of  $A$  and  $\hat{A}$  are sparse, but a good approximate inverse of  $A$  is double irregular sparse, whose numbers of irregular columns and rows are no less than those of  $A$ . Moreover, for the double regular sparse  $\hat{A}$ , we find that a good sparse approximate inverse of  $\hat{A}$  is double regular sparse and is considerably sparser than that of  $A$ . These observations imply that SPAI is not only costly to implement but also cannot construct effective

TABLE 5.2

Some information on  $A$  and  $\hat{A}$ , where  $s_1$  and  $s_2$  denote the number of irregular columns and rows, respectively,  $p = \lfloor \frac{\text{nnz}(A)}{n} \rfloor$ ,  $p_{dc}$  the number of nonzero entries in the densest column of  $A$ ,  $p_{dr}$  the number of nonzero entries in the densest row of  $\hat{A}$ , and  $\nu = \max_{1 \leq i \leq s_2} \|V_2(:, i)\|$ . The Matlab function `condest` is used to estimate the 1-norm condition numbers of the last six larger matrices

matrices	$s_1$	$s_2$	$p$	$p_{dc}$	$p_{dr}$	$\text{nnz}(\hat{A})$	$\nu$	$\kappa(A)$	$\kappa(\hat{A})$
rajat04	5	6	8	642	659	5702	$6.33 \times 10^3$	$1.64 \times 10^8$	$6.85 \times 10^7$
rajat12	9	9	6	1195	1190	6803	$2.26 \times 10^2$	$6.91 \times 10^5$	$6.36 \times 10^5$
rajat13	29	29	6	5412	5383	27232	$3.08 \times 10^3$	$1.19 \times 10^{11}$	$3.59 \times 10^8$
memplus	144	124	6	353	319	67649	$1.45 \times 10^{-1}$	$1.29 \times 10^5$	$1.24 \times 10^5$
ASIC_100k	132	129	9	92258	92165	543876	$3.93 \times 10^1$	$1.46 \times 10^{11}$	$9.27 \times 10^9$
dc1	55	54	6	114174	114184	425819	$5.59 \times 10^4$	$1.01 \times 10^{10}$	$2.21 \times 10^8$
dc2	55	54	6	114174	114184	425819	$5.74 \times 10^4$	$8.86 \times 10^9$	$5.89 \times 10^7$
dc3	55	54	6	114174	114184	425819	$6.00 \times 10^4$	$1.16 \times 10^{10}$	$1.53 \times 10^8$
trans4	55	54	6	114174	114184	425819	$1.91 \times 10^4$	$3.30 \times 10^9$	$3.46 \times 10^8$
trans5	55	54	6	114174	114184	425819	$6.01 \times 10^3$	$2.32 \times 10^9$	$6.54 \times 10^7$

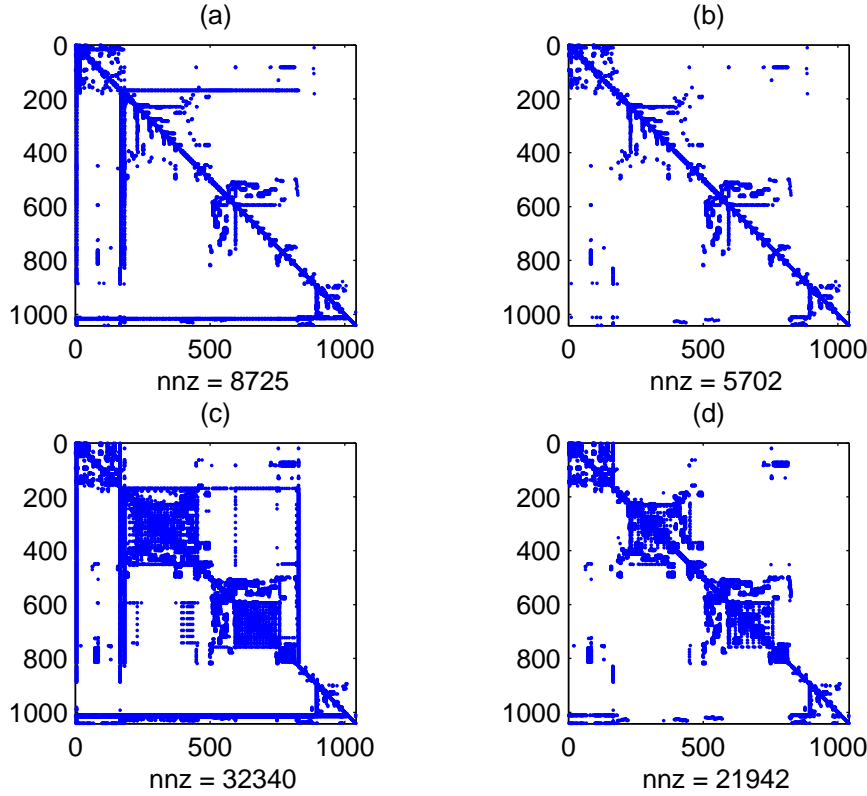


FIG. 5.1. (a): the sparsity pattern of  $A$ ; (b): the sparsity pattern of  $\hat{A}$ ; (c): the sparsity pattern of sparsified  $A^{-1}$ ; (d): the sparsity pattern of sparsified  $\hat{A}^{-1}$ .

preconditioners  $M$  of (1.1) since the  $M$  by SPAI are column regular sparse unless the loops  $l_{\max}$  are allowed to be very big, which is prohibited for SPAI. In contrast,

since good sparse approximate inverses of  $\hat{A}$  are generally double regular sparse, it is expected that SPAI, PSAI(*tol*) and RSAI(*tol*) construct good approximate inverses  $M$  of  $\hat{A}$  efficiently.

In later tables,  $spar = \frac{nnz(M)}{nnz(A)}$  or  $\frac{nnz(M)}{nnz(\hat{A})}$  denotes the sparsity of  $M$  relative to  $A$  or  $\hat{A}$ ,  $n_c$  stands for the number of columns of  $M$  whose residual norms do not drop below the prescribed accuracy  $\eta$ , and  $ptime$  and  $stime$  are the CPU timings (in seconds) of constructing  $M$  and of solving the preconditioned linear systems by BiCGStab, respectively. The notation  $\dagger$  means that we do not count CPU timings when BiCGStab fails to converge within 1000 iterations. Let the actual relative residual norm  $r_{actual} = a \cdot \varepsilon$ . We list  $a$  in the tables, where  $a < 1$  means that the relative residual norm (5.1) satisfies the prescribed accuracy  $\varepsilon$ .

**5.1. S-SPAI and N-SPAI.** In this section, we illustrate that N-SPAI is much more efficient than S-SPAI with the same parameters used in SPAI, in which the initial pattern of  $M$  is that of  $I$ , and we take  $\eta = 0.4$ ,  $l_{\max} = 20$  and add five most profitable indices to the pattern of  $m_k$  at each loop for  $k = 1, 2, \dots, n$ . The number of nonzero entries in  $m_k$  is therefore no more than  $1 + 5 \times 20 = 101$  for the given parameters. Consequently, if good preconditioners have at least one column whose number of nonzero entries is bigger than 101, SPAI may be ineffective for preconditioning (1.1). Table 5.3 shows the results, where the notation  $*$  indicates that S-SPAI could not construct  $M$  within 100 hours, for S-SPAI the notation *iter* stands for the number of iterations that BiCGStab uses, and for N-SPAI *iter* stands for the maximum iterations of BiCGStab for the  $s_1 + s_2 + 1$  systems (3.9), (3.10) and (3.11) preconditioned by  $M$ .

TABLE 5.3  
S-SPAI versus N-SPAI

matrices	S-SPAI						N-SPAI					
	<i>spar</i>	<i>ptime</i>	$n_c$	$a$	<i>iter</i>	<i>stime</i>	<i>spar</i>	<i>ptime</i>	$n_c$	$a$	<i>iter</i>	<i>stime</i>
rajat04	0.37	1.18	6	0.33	30	0.14	0.39	0.05	2	0.27	15	0.07
rajat12	0.89	3.19	3	0.58	46	0.04	0.81	0.03	0	0.35	38	0.25
rajat13	0.92	271.2	6	0.91	73	0.19	1.16	0.27	2	0.12	8	0.35
memplus	1.05	13.6	0	0.70	92	0.40	1.35	0.48	0	0.50	23	4.06
ASIC_100k	*	*	*	*	*	*	0.55	4.36	12	0.35	9	27.0
dc1	*	*	*	*	*	*	1.10	8.80	6	0.31	314	95.8
dc2	*	*	*	*	*	*	1.04	7.70	0	0.40	107	51.6
dc3	*	*	*	*	*	*	1.06	8.01	6	0.28	81	55.8
trans4	*	*	*	*	*	*	1.14	8.05	0	0.26	37	10.0
trans5	*	*	*	*	*	*	1.13	9.00	0	0.47	79	20.1

From Table 5.3, we find that all the  $a < 1$  and are of  $O(1)$  for all the test problems. It means that our replacements for  $c_0, c_1$  and  $c_2$  defined previously are good choices and work reliably and properly in practice. For the last six larger problems S-SPAI could not construct  $M$  within 100 hours but N-SPAI does the job in no more than nine seconds and BiCGStab solves all the preconditioned double regular sparse linear systems using only 10.1 to 96 seconds, very dramatic improvements! The reason is that since each of these matrices contains some columns and rows that are fully dense, SPAI spends unaffordable time in finding most profitable indices because of the large cardinalities of  $\hat{\mathcal{J}}_k$  and  $\mathcal{N}_k$  for  $k = 1, 2, \dots, n$  at each loop. Precisely, for a double irregular sparse matrix whose irregular column and row are fully dense, at each loop  $l$ , SPAI has to compute almost  $n$  numbers  $\rho_j$  by (2.8), then sort almost  $n$  indices in  $\hat{\mathcal{J}}_k$  by comparing the sizes of  $\rho_j$  and finally pick up a few most profitable indices among them. This can make SPAI fatally slow. In contrast, N-SPAI overcomes this



drawback very well since there are only very few elements in  $\hat{\mathcal{J}}_k$  and N-SPAI only needs to compute very few  $\rho_j$  and select the most profitable indices. This is why N-SPAI outperforms S-SPAI so dramatically. For each of the smaller rajat04, rajat12, rajat13 and the relatively large memplus whose irregular rows and columns are far from fully dense, we observe that the *ptime* by N-SPAI is much smaller than the corresponding one by S-SPAI by tens to hundreds of times. So, even for not very large problems, N-SPAI exhibits its much higher efficiency than S-SPAI.

We now look at the preconditioning effectiveness of  $M$ . We observe that the  $n_c$  in S-SPAI are bigger than those in N-SPAI for rajat04, rajat12, rajat13 and memplus. This demonstrates that SPAI is difficult to capture a good approximate sparsity pattern of  $A^{-1}$  when  $A$  is double irregular sparse. This is also confirmed by the numbers *iter*, which, except for rajat12, shows that BiCGStab converges much faster for the the double regular sparse problems than for (1.1) and SPAI is considerably less effective for preconditioning double irregular sparse linear systems than it is for double regular sparse ones.

Finally, we compare our transformation approach with that in [25], where double irregular sparse matrices are only transformed into column regular sparse ones, which are still row irregular sparse for our test matrices. Therefore, SPAI is still time consuming, though its efficiency is improved greatly relative to its application to  $A$  directly. We mention that for numerical experiments we use the same computer as that in [25]. Precisely, for the last six larger matrices, the approach used in [25] takes about half an hour to two hours to construct the  $M$ , while our approach here only costs about eight seconds, improvements of hundreds of times!

**5.2. S-PSAI(*tol*) and N-PSAI(*tol*).** In this section, we demonstrate that N-PSAI(*tol*) is much more efficient than S-PSAI(*tol*), where we take  $\eta = 0.4$  and  $l_{\max} = 10$ . Table 5.4 shows the results, where the notation – indicates that our computer is out of memory when constructing  $M$  due to the appearance of large sized LS problems.

TABLE 5.4  
S-PSAI(*tol*) versus N-PSAI(*tol*)

matrices	S-PSAI( <i>tol</i> )						N-PSAI( <i>tol</i> )					
	<i>spar</i>	<i>ptime</i>	$n_c$	$a$	<i>iter</i>	<i>stime</i>	<i>spar</i>	<i>ptime</i>	$n_c$	$a$	<i>iter</i>	<i>stime</i>
rajat04	0.72	1.40	0	0.79	11	0.01	0.39	0.18	0	0.15	13	0.08
rajat12	2.22	3.48	0	0.74	32	0.03	2.11	0.80	0	0.28	37	0.19
rajat13	1.36	204	0	0.19	4	0.07	0.91	0.17	0	0.29	6	0.27
memplus	6.25	624	0	0.98	215	1.30	1.78	48.6	0	0.35	27	4.08
ASIC_100k	–	–	–	–	–	–	0.43	582	0	0.34	8	27.7
dc1	–	–	–	–	–	–	1.67	1340	0	0.26	379	101
dc2	–	–	–	–	–	–	1.71	1295	0	0.36	58	48.3
dc3	–	–	–	–	–	–	1.73	1299	0	0.47	54	44.0
trans4	–	–	–	–	–	–	1.80	1679	0	0.26	28	10.9
trans5	–	–	–	–	–	–	1.62	1475	0	0.73	60	17.8

From the table, we observe that all the  $a < 1$  and are of  $O(1)$  for all the test problems. This indicates that our replacements for  $c_0, c_1$  and  $c_2$  are good choices and work reliably and properly. The table tells us that for the last six larger problems S-PSAI(*tol*) could not construct  $M$ . This is because that each  $A$  has fully dense columns, which lead to some large LS problems (2.3) with dimension  $p_{dc} \approx n$  and some fully dense  $m_k$  before dropping small nonzero entries in them; see [25, 26] for details on why it is the case. So, for  $n$  large, PSAI(*tol*) faces a severe difficulty when applied to column irregular sparse matrices directly. In contrast, with the same

parameters, N-PSAI( $tol$ ) overcomes this difficulty very well and computes effective preconditioners  $M$  efficiently for the last six larger problems. For rajat04, rajat12, rajat13 and memplus, N-PSAI( $tol$ ) constructs the  $M$  several times faster than S-PSAI( $tol$ ). However, unlike SPAI, we observe all  $n_c = 0$ , indicating that both S-PSAI( $tol$ ) and N-PSAI( $tol$ ) succeed in finding effective sparse approximate inverses, which confirms the theory that PSAI( $tol$ ) can capture effective approximate sparsity patterns of the inverse of a sparse matrix, independent of whether the matrix is regular or irregular sparse.

Next, we compare our transformation approach with that used in [25], which only transforms double irregular sparse  $A$  into column regular sparse ones. We point out that good sparse approximate inverses of a row irregular sparse matrix may have some columns which are not as sparse as a regular column of  $A$ ; see Figure 5.2 of [25] for rajat04. In this case, PSAI( $tol$ ) needs to solve some LS problems whose sizes are not as small as those for double regular sparse matrices, which causes that PSAI( $tol$ ) may be considerably more costly for only column regular sparse matrices than for double regular sparse ones. Indeed, as we have seen, for the last six larger matrices our approach costs about half of the corresponding execution time  $ptime$  used by the approach in [25].

Finally, we make some comments on N-SPAI and N-PSAI( $tol$ ). From Tables 5.3–5.4, we find that the  $spar$  by N-SPAI and N-PSAI( $tol$ ) are correspondingly comparable but N-PSAI( $tol$ ) is at least competitive with and considerably more effective than N-SPAI for preconditioning half of the test problems, as indicated by the corresponding  $n_c$ ,  $iter$  and  $stime$ . This justifies that PSAI( $tol$ ) is more effective than SPAI to capture good sparsity patterns of the inverses even for double regular sparse matrices. In addition, we have noted that N-PSAI( $tol$ ) is more costly than N-SPAI. This is simply due to our non-optimized code of PSAI( $tol$ ) in the Matlab language, whose efficiency is inferior to the optimized SPAI code written in C/MPI.

**5.3. S-RSAI( $tol$ ) and N-RSAI( $tol$ ).** In this section, we demonstrate that N-RSAI( $tol$ ) is much more efficient than S-RSAI( $tol$ ). We take  $\eta = 0.4$  and  $l_{\max} = 10$ , and use the dominant indices  $i$  with the first *three* largest  $|r_k(i)|$  at each loop for  $k = 1, 2, \dots, n$ . Table 5.5 lists the results, where the notation  $*$  for the rajat13 indicates that S-RSAI( $tol$ ) could not construct  $M$  within 25 hours and the notation  $-$  that our computer is out of memory when constructing  $M$  due to the appearance of large sized LS problems resulting from the relatively dense rows of  $A$ .

TABLE 5.5  
S-RSAI( $tol$ ) versus N-RSAI( $tol$ )

matrices	S-RSAI( $tol$ )						N-RSAI( $tol$ )					
	$spar$	$ptime$	$n_c$	$a$	$iter$	$stime$	$spar$	$ptime$	$n_c$	$a$	$iter$	$stime$
rajat04	7.26	9.05	3	0.34	17	0.05	0.52	0.31	3	0.31	19	0.09
rajat12	129	110	3	0.37	11	0.16	2.06	1.05	0	0.18	8	0.09
rajat13	*	*	*	*	*	*	3.85	13.6	2	0.12	8	0.39
memplus	8.81	675	0	0.63	16	0.18	1.73	39.0	0	0.53	16	3.09
ASIC_100k	–	–	–	–	–	–	1.54	1292	0	0.27	8	35.4
dc1	–	–	–	–	–	–	1.19	1234	2	0.29	303	80.6
dc2	–	–	–	–	–	–	1.20	1206	2	0.51	66	54.9
dc3	–	–	–	–	–	–	1.22	1311	6	0.33	72	49.1
trans4	–	–	–	–	–	–	2.30	1984	0	0.23	13	8.51
trans5	–	–	–	–	–	–	2.14	1681	0	0.22	11	11.2

We see that all the  $a < 1$  and are of  $O(1)$  for all the test problems. It is observed

from the table that for the last six larger problems S-RSAI( $tol$ ) failed to get  $M$ . This is because each  $A$  has some fully dense rows, RSAI( $tol$ ) involves solutions of some large LS problems (2.3) with dimensions  $p_{dr} \approx n$  at each loop, which exceeds the storage of our computer. However, with the same parameters, N-RSAI( $tol$ ) computes the  $M$  for double regular sparse matrices very efficiently and costs comparable CPU time to N-PSAI( $tol$ ) except rajat13. Meanwhile, these  $M$  are effective for preconditioning and have similar effects to those obtained by N-PSAI( $tol$ ), as shown by the corresponding  $n_c$ ,  $iter$  and  $stime$ . For rajat04, rajat12 and memplus, the  $M$  constructed by S-RSAI( $tol$ ) are much denser than those by N-RSAI( $tol$ ), so that S-RSAI( $tol$ ) costs much more time than N-RSAI( $tol$ ), but the  $M$  by S-RSAI( $tol$ ) have comparable preconditioning quality to those by N-RSAI( $tol$ ). For rajat13, we see that S-RSAI( $tol$ ) failed to construct  $M$  within 25 hours. This is because there is one important index  $i$  of  $r_k(i)$  that corresponds to a dense row of  $A$  at some loop when computing  $m_k$  for  $k = 1, 2, \dots, n$ , so that totally  $n$  large LS problems with dimensions comparable to  $n$  emerge and are required to solve for computing  $M$ . This is a huge computational task. Actually, we have found that S-RSAI( $tol$ ) costs about 81 hours to solve this problem. In contrast, N-RSAI( $tol$ ) is very efficient and used only 13 seconds to construct  $M$  and totally 14 seconds to solve the problem, improvements of hundreds of times!

**6. Conclusions.** In this paper, we have considered three adaptive sparse approximate inverse preconditioning procedures SPAI, PSAI( $tol$ ) and RSAI( $tol$ ). We have shown that they are very costly and even impractical for double irregular sparse linear systems, which are common in practical applications, but much cheaper to construct effective preconditioners for double regular sparse ones. To fully exploit these three preconditioning procedures, motivated by the transformation approach in [25], by making use of the Sherman-Morrison-Woodbury formula, we have proposed a transformation approach that transforms a double irregular sparse problem into some double irregular sparse ones with the same coefficient matrix. As a result, we can use the three procedures to efficiently construct good preconditioners for these double regular sparse problems and then solve the preconditioned linear systems by Krylov iterative solvers. To develop the transformation approach into a practical and reliable algorithm, we have given a number of theoretical and practical considerations. The numerical experiments have demonstrated that the proposed algorithm exhibits the very sharp superiority to the commonly used algorithm that first preconditions the double irregular sparse problem (1.1) by SPAI, PSAI( $tol$ ) or RSAI( $tol$ ) and then solves the preconditioned problem by Krylov solvers. The experiments have also illustrated that our transformation algorithm greatly improves the efficiency of the algorithm proposed in [25].

## REFERENCES

- [1] S. T. BARNARD, O. BRÖKER, M. J. GROTE AND M. HAGEMANN, *SPAI 3.2*, <http://www.computational.unibas.ch/software/spai> (2006).
- [2] S. T. BARNARD AND M. J. GROTE, *A block version of the SPAI preconditioner*, in Proceedings of the 9th SIAM conference on Parallel Processing for Scientific Computing, Hendrickson BA et al. (eds.), SIAM, Philadelphia, PA, 1999.
- [3] M. W. BENSON AND P. O. FREDERICKSON, *Iterative solution of large sparse linear systems in certain multidimensional approximation problems*, Util. Math., 22 (1982), pp. 127–140.
- [4] M. W. BENSON, J. KRETTMANN AND M. WRIGHT, *Parallel algorithms for the solution of certain large sparse linear systems*, Int. J. Comput. Math., 16 (1984), pp. 245–260.
- [5] M. BENZI, *Preconditioning techniques for large linear systems: a survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

- [6] M. BENZI, C. MEYER AND M. TÛMA, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.
- [7] M. BENZI, R. KOUHIA AND M. TÛMA, *Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 6533–6554.
- [8] M. BENZI AND M. TÛMA, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
- [9] R. BRU, J. MARÍN, J. MAS AND M. TÛMA, *Balanced incomplete factorization*, SIAM J. Sci. Comput., 30 (2008), pp. 2302–2318.
- [10] R. BRU, J. MARÍN, J. MAS AND M. TÛMA, *Improved balanced incomplete factorization*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2431–2452.
- [11] E. CHOW, *A priori sparsity pattern for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.
- [12] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Soft., 38 (2011), pp. 1–25.
- [13] I. S. DUFF, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Softw., 7 (1981), pp. 315–330.
- [14] M. FERRONATO, *Preconditioning for sparse linear systems at the dawn of the 21st century: history, current developments, and future perspectives*, ISRN Appl. Math., Vol. 2012, Article ID 127647, 49 pages, doi:10.5402/2012/127647.
- [15] J. R. GILBERT, *Predicting structure in sparse matrix computations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 62–79.
- [16] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 3rd ed., John Hopkins University Press, Baltimore, MD, 1996.
- [17] M. J. GROTE AND T. HUCKLE, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
- [18] T. HUCKLE, *Approximate sparsity pattern for the inverse of a matrix and preconditioning*, Appl. Numer. Math., 30 (1999), pp. 291–303.
- [19] C. JANNA, N. CASTELLETTO AND M. FERRONATO, *The effect of graph partitioning techniques on parallel block FSAI preconditioning: a computational study*, Numer. Algor., 68 (2015), pp. 813–836.
- [20] C. JANNA AND M. FERRONATO, *Adaptive pattern research for block FSAI preconditioning*, SIAM J. Sci. Comput., 33 (2011), pp. 3357–3380.
- [21] C. JANNA, M. FERRONATO AND G. GAMBOLATI, *A block FSAI-IIU parallel preconditioner for symmetric positive definite linear systems*, SIAM J. Sci. Comput., 32 (2010), pp. 2468–2484.
- [22] C. JANNA, M. FERRONATO AND G. GAMBOLATI, *Enhanced block FSAI preconditioning using domain decomposition techniques*, SIAM J. Sci. Comput., 35 (2013), pp. 229–249.
- [23] Z. JIA AND W. KANG, *A residual based sparse approximate inverse preconditioning procedure for large sparse linear systems*, arXiv: math.NA/1503.04500, 2015.
- [24] Z. JIA AND Q. ZHANG, *Robust dropping criteria for F-norm minimization based sparse approximate inverse preconditioning*, BIT Numer. Math., 53 (2013), pp. 959–985.
- [25] Z. JIA AND Q. ZHANG, *An approach to making SPAI and PSAI preconditioning effective for large irregular sparse linear systems*, SIAM J. Sci. Comput., 35 (2013), pp. A1903–A1927.
- [26] Z. JIA AND B. ZHU, *A power sparse approximate inverse preconditioning procedure for large sparse linear systems*, Numer. Linear Algebra Appl., 16 (2009), pp. 259–299.
- [27] L. YU. KOLOTILIAN AND A. YU. YEREMIN, *Factorized sparse approximate inverse preconditionings I. theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
- [28] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, PA, 2003.